AD-A253 351

Technical Report

CMU/SEI-92-TR-3
ESD-92-TR-3

(2)

Carnegie-Mellon University
Software Engineering Institute

An Analysis Technique for Examining Integration in a Project Support Environment

Alan W. Brown
Peter H. Feiler

January 1992

Technical Report
CMU/SEI-92-TR-3
ESD-92-TR-3
January 1992

An Analysis Technique for Examining Integration in a Project Support Environment

Alan W. Brown
Peter H. Feiler
Software Development Environments Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

John S. Herman, Capt, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

Table of Contents

List of Figures

An Analysis Technique for Examining Integration in a Project Support Environment

Abstract: In this paper we describe the use of a Project Support Environment (PSE) services reference model as an analysis technique that helps in describing, understanding, and comparing aspects of integration in a PSE. The model is briefly described, before being used as the basis for discussing a number of issues with regard to PSE integration. A major focus of the paper is a discussion of the interfaces of interest in a PSE—interfaces within a single service, between services, and between services and the PSE end-users.

The paper concludes with a discussion of possible interpretations and developments of the model to suit different user requirements.

1. Introduction

Understanding the integration characteristics of a Project Support Environment (PSE) is an important requirement for anyone engaged in the task of describing, selecting, or comparing PSE products and tools. However, while a number of attempts have been made at trying to define the meaning of the term "integration" as it applies to a PSE, there has been little written about the way in which PSEs can be described and compared with regard to their integration characteristics.

In this paper we present an analytical technique for examining the integration needs of a collection of tools with respect to a PSE. The technique is based on the application of a PSE reference model that characterizes a PSE in terms of the services it provides to its users. A companion paper to this describes the conceptual basis for such a PSE services reference model [1]. The reference model itself is being developed by the Project Support Environment Standards Workings Group (PSESWG) of the U.S. Navy's Next Generation Computing Resources (NGCR) program. They have developed their model based on that conceptual basis, expanding, clarifying and amending the concepts to suit the particular aims of NGCR. The result of their work is a detailed description of a PSE services reference model [5].

In this paper, we look at how a services-based PSE reference model can be used to gain a deeper understanding of the key issue of integration in a PSE. In particular, we look at the way in which existing tools and systems can be characterized in terms of the reference model. Different ways of carrying out the characterization are described, each aimed at providing different kinds of insight into the system. For example, a characterization aimed at understanding how well a tool matches a user's development process will be significantly different from a characterization aimed at understanding what internal mechanisms it employs. A key aspect of the use of a PSE reference model is that it facilitates this range of different uses.

The relationship between services and interfaces in a PSE reference model is of particular interest. The services provided in a PSE must be made available to, and accessed by, end-users, tools, and other services. This is the role of a service interface. By analogy to a reference model developed some years ago in the database area, we discuss in this paper

how identification of PSE services provides an essential stepping stone to the identification of relevant interfaces.

The remainder of the paper is organized as follows: Section 2 provides an overview of a PSE services reference model, with sufficient detail to allow the application of the reference model to be understood. Section 3 describes a number of ways in which the model can be used to characterize tools and systems. Section 4 focuses on the identification of interfaces, providing some examples to show how the model explains the notion of integration in a PSE. Further issues of interpreting and extending the model are examined in Section 5, before the paper concludes with a summary in Section 6.

2. A Summary of the Reference Model

Identification of services is the basis of a PSE services reference model. Services represent aspects of the functionality of a PSE as they are perceived by different PSE users. Describing a PSE in terms of services abstracts away from many aspects of particular implementation architectures, allowing a general understanding of a PSE to be defined that is not based on particular tools or technologies.

A PSE services reference model would be of little practical value if it simply provided a set of service descriptions. Some structuring of those services is required to provide a better understanding of how the services are related to each other and to the different kinds of PSE end-users.

Rather than impose a service structuring based on a particular tool, system, or standard, the PSE services reference model employs a structuring based on recognizing three levels of abstraction within the services—process, end-user, and infrastructure mechanisms—as illustrated in Figure 2-1. The end-user services describe the functionality available to PSE

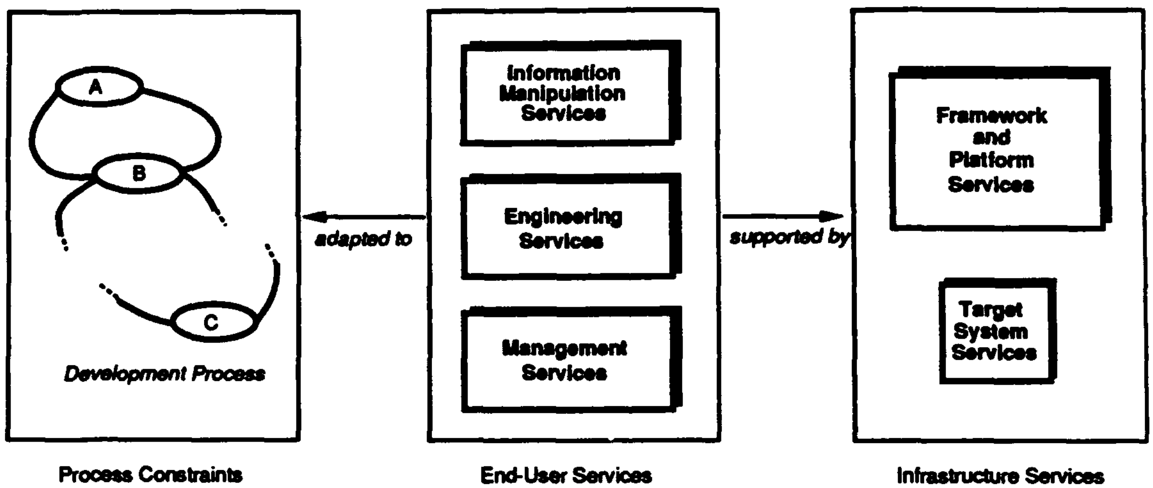

Figure 2-1 The Main Components of the PSE Services Reference Model

end-users in a populated environment. These are typically provided by the tools that populate a PSE. The infrastructure services describe the mechanisms that support those end-user services. Such services are typically a consequence of the platform on which the PSE is hosted and any additional framework mechanisms employed. End-users interact with services within the constraints imposed by a particular systems development process. Hence, that process acts as a filter on the end-user services of a PSE, guiding and constraining allowable uses, interconnections, and interfaces between end-user services. Elements of any particular PSE service can typically be described at all three of these levels. For example, configuration management services in a PSE may require infrastructure mechanisms to maintain multiple copies of objects, end-user services to provide check-in and check-out operations for system components, and policy constraints to ensure that component update is coordinated within

the development process.

Also illustrated in Figure 2-1 is a possible partitioning of both the end-user services and infrastructure services. For example, in the infrastructure services it is often convenient to distinguish those services expected as part of the computing platform on which a PSE operates (such as operating system services), and those specific to the PSE framework (such as an Object Management System (OMS)). The partitioning of services can take place at even finer levels of granularity if desired. For example, the framework services can be partitioned into OMS, process, communications, user interface, and administration services, following the work of the National Institute of Standards (NIST) and the European Computer Manufacturers (ECMA) on a CASE environment frameworks reference model [3]. A partitioning of the PSE services is provided to this level in a companion paper by Brown and Feiler [1], while a full description of the services in the reference model is available in the reference model document being developed by NGCR PSESWG [5].

In summary, a PSE services reference model describes a PSE as a collection of services, distinguishing between services directly available to the PSE end-user and those provided internally as part of the PSE infrastructure. Furthermore, policy aspects of the PSE based on constraining the use of end-user services to a particular development style are considered as external to these service descriptions, and can be defined separately. Hence, the reference model has the following characteristics:

- It expands on the NIST/ECMA reference model to address a complete PSE (i.e., an environment populated with tools) rather than restricting itself to the framework aspects of a PSE.
- The overall approach of the reference model is sufficiently generic that it can be applied to a number of different engineering domains. Software engineering environments (SEEs), although the particular interest of our work, represent just one of these engineering domains. Other domains include electrical engineering (or electronic computer-aided design (ECAD)), mechanical engineering, and manufacturing engineering.
- The reference model does not presuppose a particular PSE architecture. In particular, as with the NIST/ECMA reference model, stating that a particular PSE tool or product "conforms" to this reference model does not mean anything. Rather, the purpose of this reference model is to characterize those tools and products, enabling a better understanding of the functionality they provide. As described in detail later in this paper, such a characterization facilitates a discussion of the ease (or otherwise) with which such products can be combined in a complete environment.
- By taking a service-based approach to defining a reference model, we abstract away from individual tool products. Hence, the reference model is independent of any single tool, and does not rely on a one-to-one mapping from services to tools. In particular, an actual PSE may be constructed with one tool providing many services, many tools providing the same service, or a single tool providing many services.

To support the characterization of particular tools and products in terms of the reference model, the notion of a *profile* can be introduced. A profile is a characterization of an actual or proposed tool, system, or standard in terms of the elements of the PSE services reference model.[1] In the remainder of this paper we make extensive use of profiling techniques in understanding and comparing PSE tools and products.

[1] In some places the term *profile* is used in a much more restricted sense, meaning a particular selection of a portfolio of standards from a set of candidates. In this paper we are using the term in a much more general sense.

3. Using the Reference Model

Having briefly reviewed the various components of the reference model, it is now necessary to discuss the relationship between those components. This is best achieved by considering different ways in which the reference model can be used, highlighting its flexibility and utility.

The relationship between reference model components is based on the idea of profiling. We can identify different kinds of profiles by considering the relationships between different elements of the reference model. Before we do this, however, we must first discuss the relationship between the reference model as a collection of abstract services, and the actual tools and systems in which we are interested. The simplest way to view this relationship is to see a tool or system as an instantiation of a collection of services. In other words, with the help of the reference model any given tool or system can identify the set of services that it provides. We can do this both at the infrastructure level (i.e., what services it implements), and at the end-user services level (i.e., what it makes available to end-users).

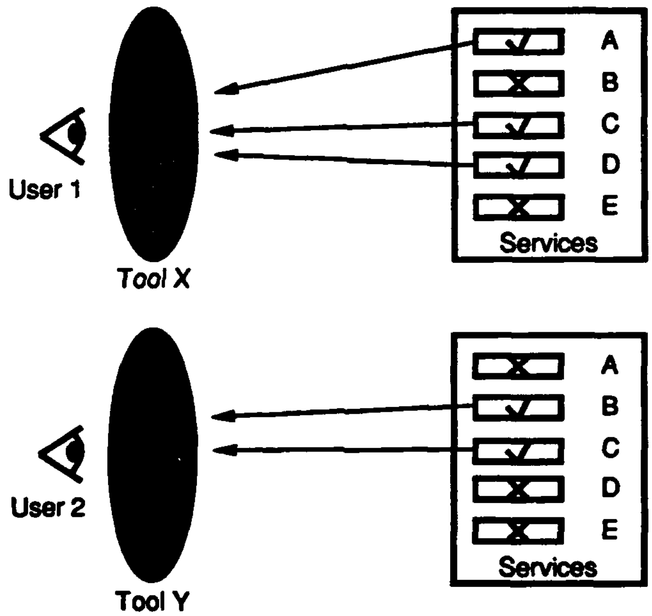

Figure 3-1 Relating Systems/Tools to Services

Figure 3-1, for example, illustrates two different tools whose services are being identified with the help of the reference model. The initial step is simply to identify which services are provided by the tool or system. For example, at the end-user services level a tool X may be found to have a design service, configuration management service, and data management service, while tool Y has a specification service and a configuration management service. Based on this initial identification of services, further analyses may take place—more detailed examination of the services provided, analysis of overlap in services between tools, discussion of relationship between end-user services and their use within a particular development process, and so on. Each of these possible analyses can be considered a different kind of

system profiling technique. In the following sections we examine a number of these profiles.

3.1. A Process View of End-User Services

A typical end-user of a PSE will perceive, and interact with, end-user services within the context of a software development process. This process enforces particular usage patterns, constrains service interconnections, and guides the user in the application of the end-user services. Hence, the only way in which an end-user makes use of the end-user services is in support of that process, as illustrated in Figure 3-2.

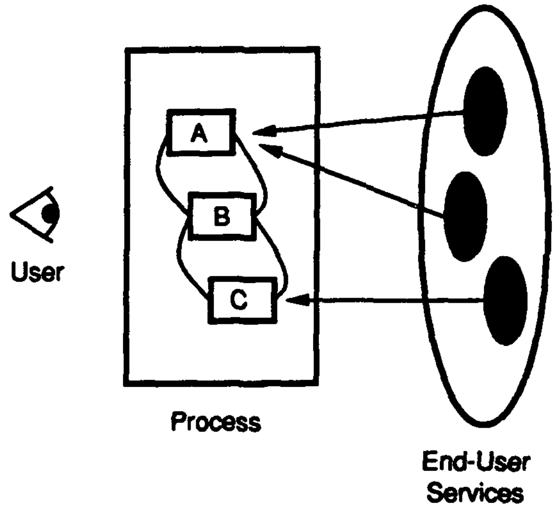

Figure 3-2 A Process View of End-User Services

The reference model allows a profile to be described in which end-user services are related to each other through a software development process that ties them together. For example, one end-user service may precede, may take place in parallel, or may follow another service in a particular software development process. This relationship between services dictates the ways in which those services must interact. The reference model allows an actual or proposed PSE product to be described in terms of the end-user services it provides and the ways in which those services are used in support of a particular software development process. The result of such a profile will be an understanding of which aspects of the software development process are supported by which PSE end-user services, together with an analysis of the interconnection of services that is required for that process. This information can be used, for example, by environment builders who need to know which tools to integrate. Rather than being in the position of integrating every tool with every other tool, taking a process view of end-user service use makes it possible to identify those services that have a direct relationship within that process. The environment builder can then focus on the tools that provide those services requiring interconnection.

3.2. Different Process Views of the End-User Services

A single PSE may be used by many people employed on different projects. Each project typically defines its own software development process, with the result that the same end-user services provided by the PSE are viewed differently by different users due to the fact that they are following different software development processes, as illustrated in Figure 3-3.

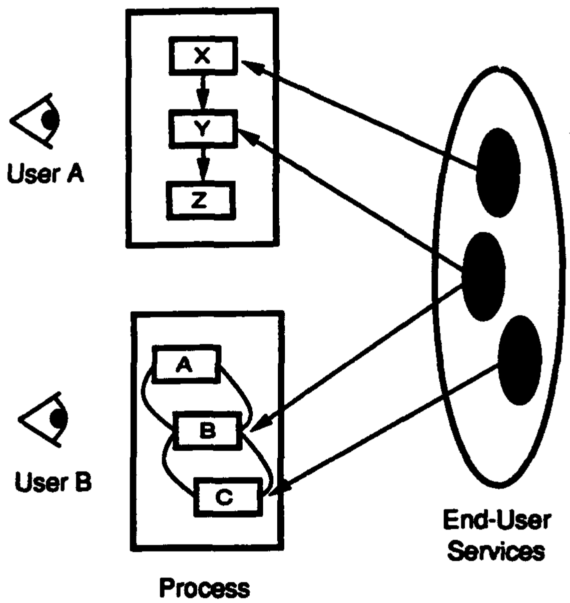

Figure 3-3 Different Process Views of the End-User Services

Using the reference model in this way allows different software development processes to be directly compared with respect to the support provided for them by a particular PSE. Similarly, when considering the purchase of a PSE, providing such a profile would help to ensure that the PSE provided sufficient support for an organization's current software development processes.

3.3. Different User Roles Within the Same Process

Even within the same software development process the end-user services are perceived differently by different users. This is due to the fact that users carry out different roles within the process—developers, managers, administrators, and so on. Each role provides a distinctive view of the end-user services of the PSE, and may be interested in different aspects of the development process, as illustrated in Figure 3-4.

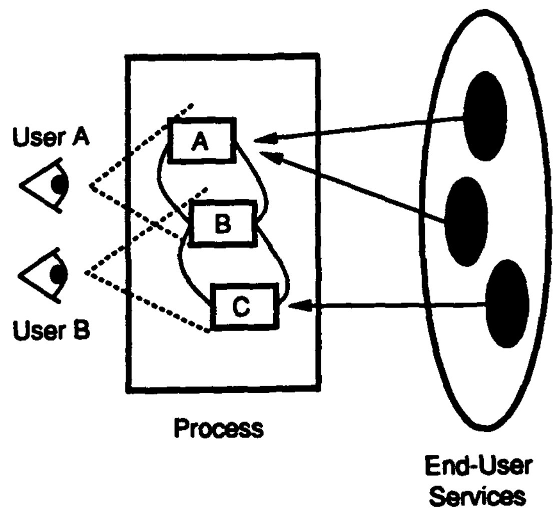

Figure 3-4 Different User Roles within the Same Process

Profiling in this way allows comparison to be made between the support provided for different roles within the software development process. For example, in many PSEs there are end-user services that are well suited to software designers and developers, but fewer such services that support quality assurance engineers and software development managers. Profiling in this way would allow the disparity to be identified.

3.4. Relating End-User Services to Their Implementation

To realize a set of end-user services, a PSE must provide mechanisms to implement them. One end-user service may have a variety of possible implementations, one infrastructure mechanism may implement many services, or one end-user service may in fact be an abstraction of many infrastructure services, as illustrated in Figure 3-5.

Typically, an end-user service may be implemented in different ways by different PSEs. Therefore, when understanding how an existing PSE is constructed, it is important to be able to relate the services as seen by the end-user to the underlying mechanisms that implement those services.

In addition to providing a characterization of end-user services in terms of infrastructure services, relationships between end-user services can be reflected in the sharing of infrastructure services. For example, the same infrastructure service may be the basis for a number of end-user services.

A useful distinction can be made between infrastructure services that are provided by a tool to support an end-user services, and those that are used by that end-user service to carry out its task. For example, a tool may implement some of its own data management services, but may assume the existence of a file system to support it. This distinction is important because

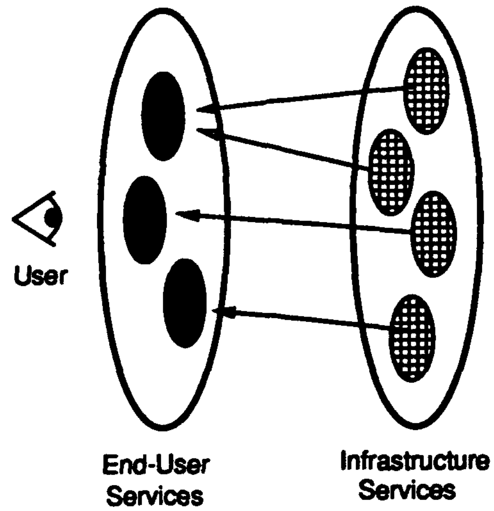

Figure 3-5 Relating End-User Services to Their Implementation

it highlights the fact that infrastructure services do not exist in isolation. It is important to know, for example, if a database mechanism (i.e., an infrastructure service) implements a data repository service (i.e., an end-user service) that the database mechanism itself makes use of a distributed filestore to record its data.

3.5. Comparing PSEs

When two different PSE products or architectures are compared or evaluated, some means of comparison is required. Such a comparison can take place at a logical level in terms of perceived functionality of the systems, and/or at a physical level in terms of implementation mechanisms, as illustrated in Figure 3-6.

The reference model provides a basis for comparing PSEs at both of the levels suggested above. At the logical level, different end-user services can be described and compared. Thus, without the need to refer to implementation details, it is possible to describe the similarity in functionality of two systems. Similarly, the infrastructure services that implement the end-user services can be described and compared. Again, if required, two systems can be compared purely in terms of the infrastructure mechanisms they provide, regardless of how those mechanisms are made available as end-user services. For a more comprehensive comparison, both of these analyses may be provided.

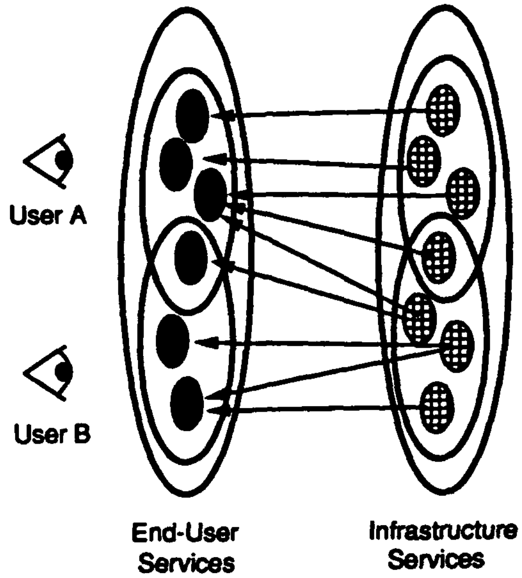

Figure 3-6 Comparing PSEs

3.6. Analyzing Implementation Mechanisms

When a tool writer wishes to integrate a tool with a PSE, both the infrastructure services of that PSE and the architecture of the tool must be fully understood. The mechanisms available, the interfaces to those mechanisms, and the best ways to combine those mechanisms in support of the tool must all be considered, as illustrated in Figure 3-7.

Typically, a PSE provides many mechanisms that may be of use to the tool writer, and there are important implications in terms of tool functionality, adaptability, and ease of communication with other tools based on the implementation decisions made by the tool writer. This profile allows integration alternatives to be more explicitly defined and described, and therefore more accessible to the tool writer.

An interesting aspect of this work is that different tool integration approaches can be evaluated with regard to their effectiveness in supporting the needs of tools. For example, even though the need for data sharing between two tools has been identified, a number of possible integration strategies are possible, such as the use of a common repository, a data interchange language, data translation via a control process, and so on. The particular tool architectures, together with the facilities available via the PSE framework and platform services, will all have an impact on the alternative chosen.

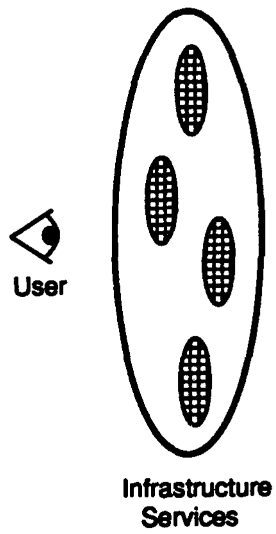

Figure 3-7 Analyzing Implementation Mechanisms

3.7. Understanding the Complete Picture

In designing, implementing, and maintaining a PSE it is often necessary to understand the relationship between many elements of the PSE architecture. We have distinguished three levels where that understanding is important, as illustrated in Figure 3-8.

Ideally, the designers of a PSE could employ the reference model as a guide in their design, ensuring that all three levels have been fully considered. The end-user services of the PSE will then be designed to meet existing end-user requirements, infrastructure mechanisms will be selected that best support those end-user services, and the end-user services will be configurable to allow adaptation to different development processes. The reference model allows many of these issues to be identified in isolation and the interdependencies to be examined within a consistent framework.

Similarly, to gain an understanding of a particular PSE product and its usefulness for supporting an organization in its software development, such a profile focuses attention at three levels, process, end-user services, and infrastructure services, allowing each level to be analyzed in isolation and the relationship between levels to be separately identified.

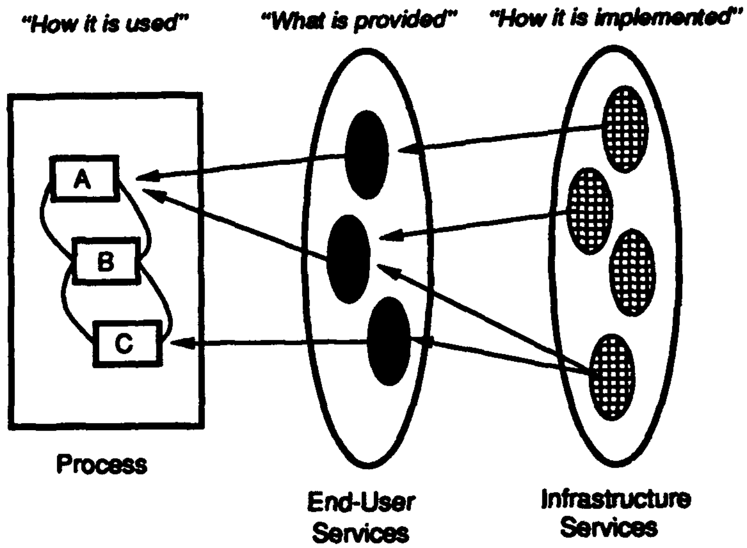

Figure 3-8 Understanding the Complete Picture

4. Services and Interfaces

One of the important aims of the reference model with regard to the NGCR PSESWG effort is to help in the identification of interfaces at which standardization is both desirable and practical. Having described a services-based PSE reference model, it may still be unclear how we are able to satisfy this aim.

We can show how the reference model provides a necessary basis for addressing this issue by considering database systems, an application area where such a reference model has been used. The database world has one of the most widely accepted reference model of any application area in the ANSI/SPARC/X3 model.

4.1. The ANSI/SPARC/X3 Model

Adopting our terminology for the ANSI/SPARC/X3 model, three levels of services were considered—the external level, conceptual level, and the physical level [6]. End-users perceive the database in terms of a set of external services. These can be different for each class of end-user. The external services are presentations of a single set of conceptual services. These provide a common basis for discussing the database at a logical level, independent of implementation considerations. Finally, the conceptual services are implemented in terms of a set of physical services. These may be specific to a particular database management system (DBMS) and/or platform.

For this reference model, a particular DBMS product can be described in terms of the three levels identified (and the relationships between those three levels). Hence, it is possible to identify and describe the conceptual level services that are provided, how they are physically realized in terms of the physical level services, and what external views of the conceptual services can be constructed to provide external services to the end-user.

Furthermore, the reference model can be used in another way by identifying the *interfaces* to those services. At each level a data definition language (DDL) and a data manipulation language (DML) provide an interface to the services. Different database products can be examined and the DDL and DML they provide at each of the three levels can be determined.

Of course, it would be helpful if different database products implemented *the same* DDL and DML at each of the three levels (i.e., if they supported the same interfaces). This would ensure that applications built on those products are portable across the different implementations. In fact, there have been attempts at standardizing the interfaces, and there are different possible standards that a database may implement at each of the three levels of this reference model. For example, the CODASYL Network Data Language (NDL) and ANSI Structured Query Language (SQL) are such standards. Both of them actually provide operations at each of the three levels, with NDL concentrating mainly on the conceptual and physical levels and SQL on the conceptual and external levels.

In summary, we see that, through the ANSI/SPARC/X3 reference model, it is possible to identify interfaces to the services provided at three levels within a database system and to consider standards as they apply to those interfaces.

4.2. Back to PSEs

Relating the database experiences to the PSE world, we see that the reference model we have proposed has many similarities to the database one. Therefore, we see that we need to:

1. Understand the services that are provided by existing and proposed PSE products at the different levels we have identified (process, end-user, and infrastructure).

2. Examine the interfaces to those services as provided by existing and proposed PSE products.

3. Analyze what is commonly provided at those interfaces, and where there is substantial agreement between products, by identifying possible standards at each level.

The PSE reference model provides a template to guide all three of the above steps. It helps to partition the services into the three levels, identify the relationships between the three levels, and finally to examine the interfaces to each level of services. By taking this approach, the proposed services-based model allows us to discuss services provided by a PSE, then to examine the interfaces to those services with the aim of identifying relevant PSE standards. Figure 4-1 illustrates this point by showing where those interfaces occur. We refer to these

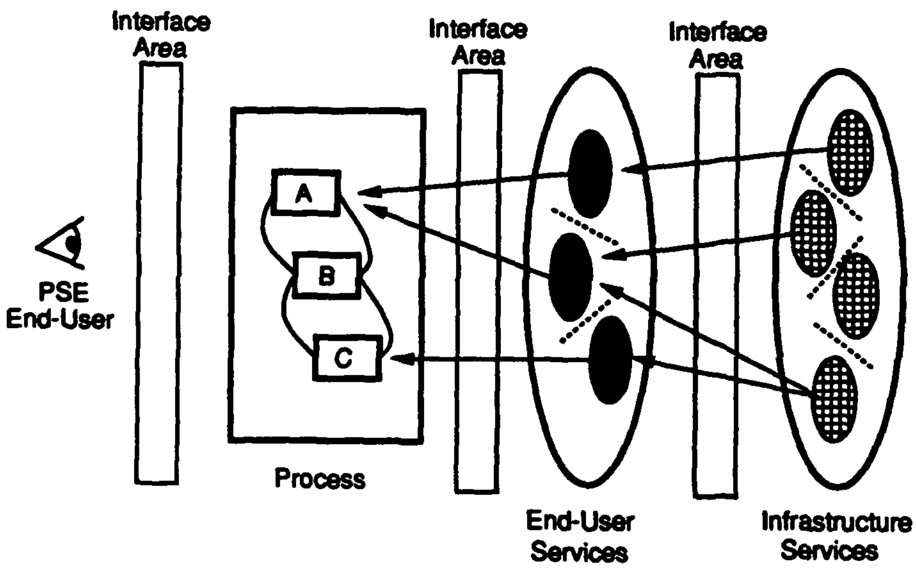

Figure 4-1 The PSE Service Reference Model with Interfaces

as *interface areas* as they may be composed of a collection of interfaces. For each of these interface areas potential standards can be identified, competing standards evaluated, and so on.

Notice that we may be interested in a number of possibly overlapping interfaces and interface standards. This is due to the wide variation in classes of PSE user (e.g., managers, software developers, test engineers, etc.), application areas for a PSE (e.g., software development, hardware development, real-time systems, data processing applications, etc.), and PSE architectures (e.g., CASE tool based, IPSE framework based, process centered, database centered, etc.). All of these result in required end-user services and infrastructure services to support the end-user services. Hence, many different standards may be possible for each interface area of interest.

Most existing work in defining PSE standards has taken place in the infrastructure services area. Relevant standards here include the work on POSIX, and the ECMA PCTE SEE framework standards. Similarly, at the process level there has been some work on defining standard life-cycle models such as the U.S. Department of Defense standard, DoD-MIL-STD-2167A. In the end-user services, however, it is much more difficult to see what standards might be appropriate. Possibilities could include data modeling standards (e.g., a standard data modeling notation such as ER diagrams), and design notation standards (e.g., use of a design method such as SSADM, mandated by the UK government). More meaningfully, what would be required at this level are common information models describing agreed data structures and their semantics in particular application domains. Such agreement in a general sense seems a long way off. However, work in three areas may be a possible basis for such agreement in the future:

1. Various small coalitions of tool vendors have been formed, realizing that the route to tool integration between their tools is to provide accessible programatic interfaces to their tools' services, and to share a common information model between their products. Application developers can then purchase a suite of tools from different vendors that share common data structures and semantics, and make use of services provided by each others tools.

2. Message-based PSE systems such as HP's SoftBench rely on classes of tools responding to, and producing, a predetermined set of messages. For example, a "check-in" message sent by a program editor is acted upon by any particular configuration management tool in the PSE. Hence, tool vendors and implementers of message-based PSE systems are now coming together to begin to discuss the definition of standard sets of messages for particular tool classes.

3. Defining and agreeing information models for particular domains is the aim of much of the work in the reuse area. They believe that reuse in any meaningful way requires a shared understanding of the context in which the reuse takes place. It may well be that this work provides insight for the PSE community.

4.3. Examples

To illustrate the use of the reference model we look at two different examples. The first example concentrates on profiling in the process and end-user services aspects of the model, the second in the end-user services and infrastructure services aspects.

4.3.1. Comparison of End-User Services

Suppose our development process was concerned with design, coding, and testing software modules. For each of these activities we may wish to obtain different tools possibly from different vendors. Clearly, there is a relationship between the three activities, reflected in interaction between the respective tools. The difficulty is in understanding, and accommodating, this tool interaction.

We can use the reference model to help us to understand the relationship between the development process and the tools, and between the tools themselves. First, we must determine which end-user services are provided by each of the three tools, as illustrated in Figure 4-2. For example, the coding tool may implement services for creating, analyzing, and translating code, as well as for managing versions of the code, team coordination, and system building.

Even at this high level, we can immediately see where overlap of end-user services between tools is taking place. For example, both the design and the coding tools provide version management services. The likelyhood is that they are *different* version management services (e.g., different naming schemes, locking policies, check-in/check-out semantics, and so on). Based on an analysis of the development process needs of the organization, one of a number of possible ways forward is possible, including:

- It may be possible for only one tool to provide the common service. For example, the coding tool could delegate version management to the design tool. In this way a single consistent approach to the common service is used.
- The overlapping services co-exist with some form of translation, or synchronization, necessary between them. For example, some code could be written to map the design tool's version naming scheme to the coding tool's version naming scheme. Manual procedures may be required to ensure versions are kept consistent.
- The overlapping services co-exist with no conflict. For example, within the development process of a particular organization it may cause no undue problems for the design tool and coding tool to maintain separate version histories. This approach will be dependent of the organization's development process needs.

There are implications on the mechanisms handling the end-user services if either of the first two options above is chosen. These can be examined separately with a clear understanding

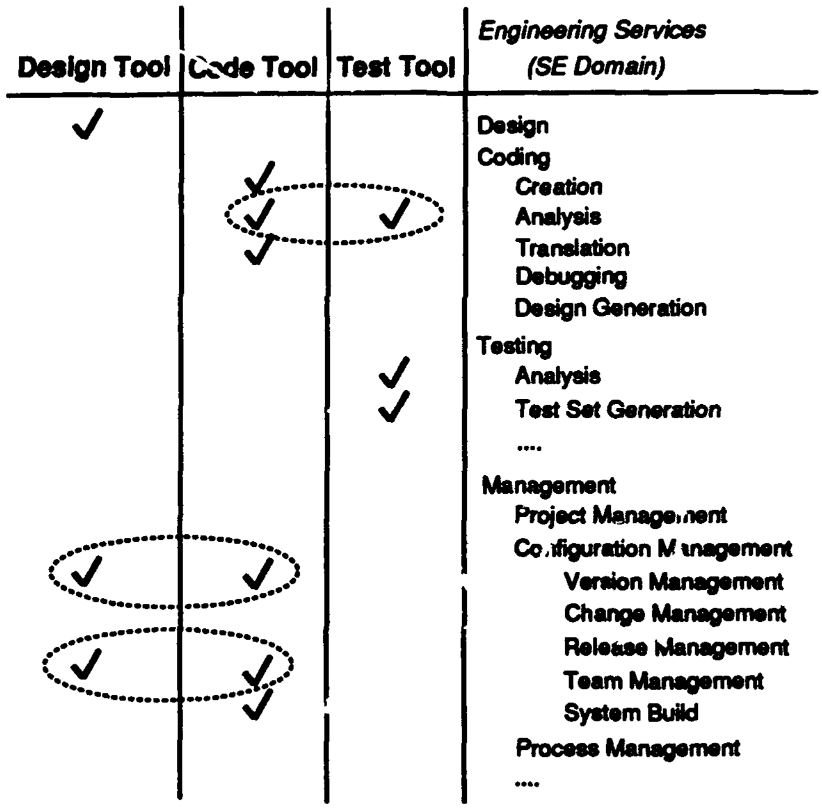

Figure 4-2 A Comparison of End-User Services from Three Tools

of the requirements of that examination. Such an analysis helps in identifying areas that are in need of further attention.

4.3.2. Different forms of Data Integration

The second example illustrates the use of this reference model in understanding different kinds of data integration in a PSE. In particular, concepts such as information model, data model, and data storage mechanisms are often confused in much existing work. However, there are important distinctions that should be made between them. Using the reference model we can highlight a number of data integration aspects and discuss the consequences of different "levels" of data integration [2].

In this reference model the end-user services describe the PSE from an end-user perspective. As such, they concentrate on the information and domain models perceived by end-users, providing a description of some part of the end-user's development process. In terms of infrastructure services, those information and domain models are represented in a particular modeling formalism such as the Entity-Relationship (ER) notation, or an object-oriented (OO) modeling notation. These, in turn, are implemented by a particular data storage mechanisms

such as an object management system (OMS), database management system (DBMS), or file system.

For example, consider the following three examples of data integration, illustrated in Figure 4-3. In the first example, two design tools have a common data model and data storage

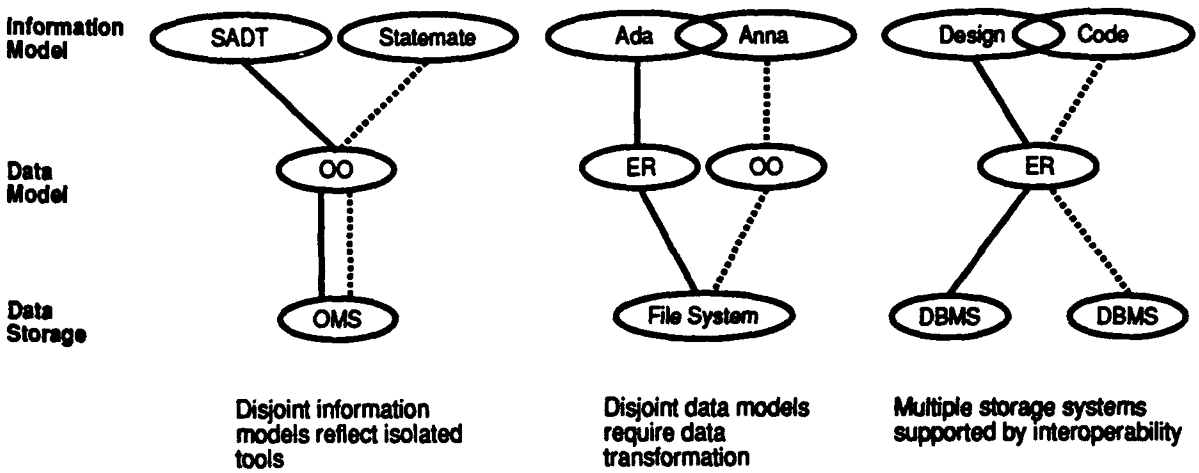

Figure 4-3 Three Different Integration Examples

mechanism. Both tools use object-oriented data modeling concepts and store their data in the same object management system. From an infrastructure point of view, they are "well integrated". However, they have different information models. For example, there may be no common understanding of the meaning of concepts such as "module", "user", "task" and so on. From an end-user services point of view, they are not well integrated because they employ different information models. Sharing of information between the tools will be difficult to achieve.

The second example shows some agreement between the information models in the Ada programming language and the Anna specification language. Both have a common understanding of Ada syntax and semantics. However, because they use different data models in their implementation, some form of data transformation is required to ensure they share the same files in the file system. Thus, the integration at the end-user services level is not matched by integration at the infrastructure level unless data transformation mechanisms are employed.

The third example looks at a design and a coding tool that have agreed on a common information model, both using an ER data model to represent that model. However, each tool has its own DBMS in which it stores that model, and therefore its data. As a result, while the two tools are well integrated at the end-user services level, mechanisms will be required to ensure synchronization and consistency across the databases to provide integration at the infrastructure services level.

Focusing on this third example in more detail, we see how the reference model has allowed us to identify the requirement for integration at the mechanism level. The next step would be

to select alternative strategies for addressing this requirement, and assessing their suitability. In the case of this example, two possible strategies are:

1. allow interoperation between the two tool DBMSs by implementing a control process that maintains consistency across the DBMSs, interprets global data queries, and so on.
2. maintain a single controlling DBMS into which the tool DBMSs copy and transfer data.

The choice between these and any other alternatives will be made based on a number of factors, including architectural considerations of the tools and the environment and performance requirements based on the expected usage patterns of the tools.

4.3.3. Summary

In summary, we can see from the simplified examples above that this PSE services reference model can be used in a variety of ways to gain a better understanding of some of the issues when considering integration of tools in a PSE. Figure 4-4 illustrates this point by summarizing a more complete context in which tools are integrated as viewed through this reference model.

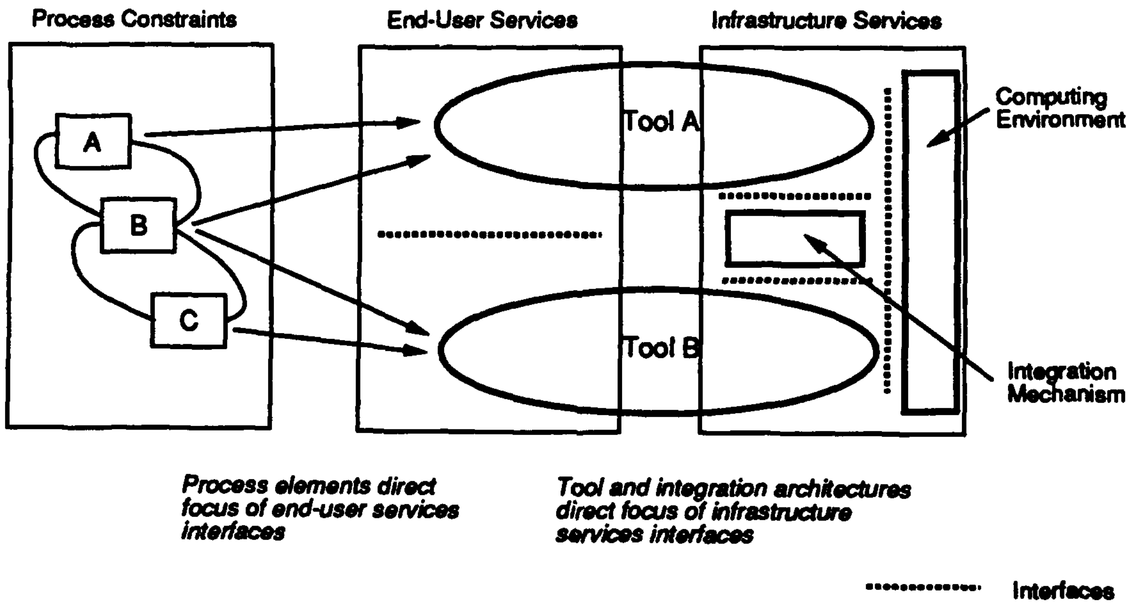

Figure 4-4 Tool Integration as Viewed Via the Reference Model

Tools consist of end-user services supported by infrastructure services. Typically, there are infrastructure services provided by the host computer environment via an operating system and a PSE framework product. To integrate two tools it is necessary to consider the infrastructure aspects of integration in terms of the way in which the tools communicate to the

operating system and framework and to each other. Similarly, the end-user services provided by the tools must be reconciled, providing semantic consistency between the objects being manipulated, perhaps through agreement of a common information or domain model.

The end-user services interface between tools can be identified through the constraints imposed by the development process in which the tools will participate. In particular, rather than having to consider the relationship between all of the services provided by the tools, the development process acts as a filter by allowing the tool integrator to focus on only those services that logically share information. For example, if the development process constrains module integration to be performed by a separate product test group, then the integration of two module development tools may be significantly simplified. It is the development process that guides and constrains integration of end-user services.

At the infrastructure services level the tools must both operate in the same computing environment (e.g., be able to operate on the same machine and operating system), and make use of particular integration mechanisms to ensure data sharing and communication between the tools. Having identified which end-user services must be integrated, the mechanistic aspects of integration can typically be carried out in a number of different ways. The architecture of the computing environment, integration mechanisms available, and the tools themselves direct the ways in which integration at this level can be implemented. Using the (partial) classification of information transfer mechanisms described in the IEEE standard reference model for computing system tool interconnection [4], we can identify four ways that tools can share data:

1. *Direct transfer* using an inter-process communication mechanism such as remote procedure calls.
2. *File-based transfer* through an intermediate data interchange language.
3. *Repository-based transfer* using a central database to store common information.
4. *Communicating system transfer* with an application-level communication mechanism such as a message broadcast system.

Alternatives such as these could be evaluated based on architectural constraints imposed by the tools and environment and with regard to performance requirements derived from the expected usage patterns of the tools.

5. Discussion

In this section of the paper we expand on the uses and usefulness of the reference model by looking at some examples of how the model may be interpreted, and how it may be further developed to suit particular uses.

5.1. Interpreting the Model

We illustrate the utility of the PSE services reference model through the use of an example. Let us suppose that we are interested in quality assurance (QA) aspects of software development and their support within a PSE. We believe there are a number of ways of using the PSE services reference model in this case, including:

- *Identification* of the end-user services that form part of the QA function. For example, as part of QA we may want to distinguish configuration management services (such as build control and version management), metrics gathering, auditing and logging, and so on, as the services we require.
- Separation of QA services from infrastructure mechanisms supporting those services. Having identified the QA end-user services of interest, these can then be mapped on to PSE *infrastructure* services that support those services. For example, underlying version control and build tools may be provided as part of the operating system facilities within the PSE platform services.
- Evaluating different PSE implementations. The QA end-user services can act as a set of requirements for evaluating different PSEs. For each PSE, the services can be analyzed according to the PSE reference model, and then compared with the required set of QA services. In addition, different PSEs may be compared and contrasted with regard to their QA support in this way.

5.2. Extending the Reference Model

It is important to recognize the need for well-defined aims for this PSE services reference model, and to evaluate its usefulness in the light of those aims. Complementing this, however, is the requirement that the model can be developed, or extended, to suit different user needs. Such extensions will allow different interpretations of the model within particular contexts. We illustrate this by looking at two possible areas in which this model could be developed further.

5.2.1. The PSE Services Reference Model and Process Issues

At the highest level, different end-user services interact and interoperate as prescribed by the end-user's software process. This interaction may be implemented in a number of different ways, each utilizing different end-user services and their interfaces.

Reconsidering the earlier example of QA, we would model process issues that define when metrics gathering takes place, how development documents gain QA approval, the relationships between the version management system and the QA department in monitoring bugs in code, and so on. Note that these are not services that are provided by the PSE—they are policy, or procedural, issues that constrain the way in which end-user services are used in support of a software development process.

Typically, for those interested in understanding and modeling process issues, some form of modeling formalism would be available. In developing the PSE services reference model, aspects of this language could be built into the process dimension based on existing and proposed standards for software development life-cycle support.

The process architecture of a PSE reflects the collection of end-user services and their interconnections that make up a particular PSE instance. It can be characterized by a particular profile of end-user service interfaces (i.e., a particular collection of domain-specific data models and behavioral models representing relevant engineering and management services). Hence, the notion of a profile can accommodate issues of process in a useful and natural way.

5.2.2. The PSE Services Reference Model and Integration

Each populated PSE has an architecture. This architecture has two components—the process architecture (as discussed above), and the integration architecture (also referred to as the environment framework architecture). The integration architecture reflects the mechanisms that have been chosen to realize end-user services and their integration. Again, a profile of infrastructure services can characterize the overall integration architecture of a PSE. For example, the profile of a central repository architecture differs from that of a decentralized repository architecture.

Typically, "real" populated PSEs have a complex structure that does not provide a single profile, but some hybrid collection of services, accessed via a number of different interfaces. Similarly, they make use of their implementation services at different levels of abstraction. For example, collections of tools within an engineering life-cycle step may be tightly integrated around a common domain data model, while coalitions of tools complement the central data storage for some information (e.g., a data dictionary) with control integration services to drive other tools. Similarly, the PSE architecture may be based on some communications model such as message passing (e.g., HP's SoftBench).

In all of these cases, we may wish to use the PSE reference model to better understand the issue of integration in a PSE. The PSE reference model can help by allowing tools to be

defined in terms of their service profiles (i.e., the services they provide). Different tools can be compared by examining and directly comparing their service profiles.

We can represent issues of integration as an integration dimension to the PSE services reference model that intersects the process, end-user, and infrastructure service levels of the model. Hence, integration of tools can be addressed at three distinct levels: at the process level the way in which the tools support policies and procedures of the end-user organization can be discussed, at the end-user services level the overlap of services provided by the tools can be analyzed, and at the infrastructure services level the mechanistic aspects of making the tools interoperate can be examined. Typically, the integration dimension would be refined with some model of integration (for example, data, process, and UI) which can act as a framework for providing a deeper understanding of those issues at the three levels.

6. Summary and Conclusions

The PSE services reference model provides a framework for describing aspects of PSE functionality independent of particular, tools, technologies, and architectures. In this paper we have taken that model and examined its usefulness in providing a better understanding of some aspects of integration in a PSE and for characterizing existing tools and systems with respect to the reference model elements.

The concept of a *profile* has been critical to this work. Profiling is the task of applying the model to a particular tool or system. Different kinds of profiles have been shown to provide insights into different aspects of a PSE. Following the numerous examples described in this paper, the flexibility and diversity of uses of the PSE services reference model have been demonstrated. In particular, we have shown that the separation of process, end-user services, and infrastructure mechanisms provides a basis for a deeper understanding of the three main activities of PSE integration:

- Selection of tools, interfaces, and standards. Ideally, these can be derived from different vendors and organizations based on the users' needs.
- Integration of those tools, interfaces, and standards to form a complete PSE.
- Adaptation and tailoring of the resultant environment to the constraints imposed by a particular development process.

In performing our examination of the use of this PSE services reference model as an analysis technique for addressing those activities, we have provided a number of examples, sufficient for illustrating various aspects of the model's use. The real test will be to take existing, complex tools and systems and to perform detailed analyses of them using the model. This is work we hope to pursue in the very near future, both within our own research and within the context of the NGCR PSESWG programme.

Acknowledgements

The roots of this work are in various environment related projects at the SEI and other places. In particular, interactions with members of the Software Development Environment Project and CASE Technology Project at the SEI, Tricia Oberndorf from NADC, Dave Carney from IDA, and participants of NIST ISEE and NGCR PSESWG have all contributed to the work reported here.

We are grateful for comments on previous drafts of this paper from Dave Carney, Ed Morris, Tricia Oberndorf, and Kurt Wallnau.

Glossary

reference model	an abstract description of a system that can act as a reference point for discussion, analysis, and comparison.
service	a collection of PSE functionality.
infrastructure	the software and hardware components of a PSE that provide the basic mechanisms for communication and coordination between tools.
end-user service	a service available for direct interaction by end-users of a PSE.
infrastructure service	a service that is provided to support, or implement, some aspect of one or more end-user services
profile	a characterization of a particular PSE tool or system in terms of the elements of the PSE services reference model.
PSE services reference model	an abstract description of a PSE based on describing a PSE as a collection of services. The description is capable of being used for discussion, analysis, and comparison of current and proposed PSE tools and products.

References

[1] A.W. Brown and P.H. Feiler. *The Conceptual Basis for a Project Support Environment Services Reference Model.* Technical Report CMU/SEI-92-TR-2, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, December 1991.

[2] A.W. Brown and J.A. McDermid. On Integration and Reuse in a Software Development Environment. In F. Long and M. Tedd, editors, *Software Engineering Environments '91*, Ellis Horwood, Chichester, England, 1991.

[3] ECMA. *A Reference Model for Frameworks of Computer-Assisted Software Engineering Environments.* Technical Report TR/55, European Computer Manufacturers Association, Geneva, Switzerland, December 1990.

[4] IEEE. A Standard Reference Model for Computing System Tool Interconnections. Technical Report Draft P1175/D11, IEEE, New York, NY, May 1991.

[5] NGCR PSESWG. A Reference Model for Project Support Environment Standards. Technical Report Draft, US Navy NGCR Programme, January 1992.

[6] D. Tsichritzis and A. Klug (eds.). The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Data Base Management Systems. *Information Systems*, 3, 1978.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-92-TR-3	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-92-TR-3
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (City, State and ZIP Code) ESD/AVS Hanscom Air Force Base, MA 01731
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESD/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO: 63756E PROJECT NO.: N/A TASK NO: N/A WORK UNIT NO.: N/A

11. TITLE (Include Security Classification)
An Analysis Technique for Examining Integration in a Project Support Environment

12. PERSONAL AUTHOR(S)
Alan W. Brown and Peter H. Feiler

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM TO	January 1992	34 pp.

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number)
integration
project support environment
services reference model

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In this paper we describe the use of a Project Support Environment (PSE) services reference model as an analysis technique that helps in describing, understanding, and comparing aspects of integration in a PSE. The model is briefly described, before being used as the basis for discussing a number of issues with regard to PSE integration. A major focus of this paper is a discussion of the interfaces of interest in a PSE - interfaces within a single service, between services, and between services and the PSE end-users.

The paper concludes with a discussion of possible interpretations and developments of the model to suit different user requirements.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPTDTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL John S. Herman, Capt, USAF	22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631 22c. OFFICE SYMBOL ESD/AVS (SEI)

ABSTRACT —continued from page one, block 19